Ćwiczenie 2

Algebra Boolea, przykłady równań logicznych.

WPROWADZENIE DO TEORII.

A. TWIERDZENIA ALGEBRY BOOLE'A

1	a	$A + B = B + A$	prawo przemienności
	b	$A \cdot B = B \cdot A$	
2	a	$A + B + C = A + (B + C) = (A + B) + C$	prawo łączności
	b	$A \cdot B \cdot C = A \cdot (B \cdot C) = (A \cdot B) \cdot C$	
3	a	$A \cdot (B + C) = A \cdot B + A \cdot C$	prawo rozdzielczości
	b	$A + B \cdot C = (A + B) \cdot (A + C)$	
4		$\overline{\overline{A}} = A$	
5	a	$A + 1 = 1$	
	b	$A \cdot 1 = A$	
6	a	$A + 0 = A$	
	b	$A \cdot 0 = 0$	
7	a	$A + A = A$	
	b	$A \cdot A = A$	
8	a	$A + \overline{A} = 1$	
	b	$A \cdot \overline{A} = 0$	
9	a	$A \cdot (A + B) = A$	
	b	$A + A \cdot B = A$	
10	a	$A \cdot (\overline{A} + B) = A \cdot B$	
	b	$A + \overline{A} \cdot B = A + B$	
11	a	$A \cdot B + A \cdot C = A \cdot (B + C)$	
	b	$(A + B) \cdot (A + C) = A + B \cdot C$	
12	a	$A \cdot B + B \cdot C + \overline{A} \cdot C = A \cdot B + \overline{A} \cdot C$	
	b	$(A + B) \cdot (B + C) \cdot (\overline{A} + C) = (A + B) \cdot (\overline{A} + C)$	
13	a	$A \cdot B + \overline{A} \cdot B = B$	
	b	$(A + B) \cdot (\overline{A} + B) = B$	
14	a	$\overline{A + B + C + \ldots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \ldots$	prawo de Morgana
	b	$\overline{A \cdot B \cdot C \cdot \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$	

B . TABLICE KARNAUGHA

1.1 Minimalizowanie funkcji logicznych za pomocą tablic Karnaugha

B \ A	0	1
0		
1		

C \ AB	00	01	11	10
0				
1				

CD \ AB	00	01	11	10
00				
01				
11				
10				

CDE \ AB	00	01	11	10
000				
001				
011				
010				
110				
111				
101				
100				

DEF \ ABC	000	001	011	010	110	111	101	100
000								
001								
011								
010								
110								
111								
101								
100								

rys. 1.1

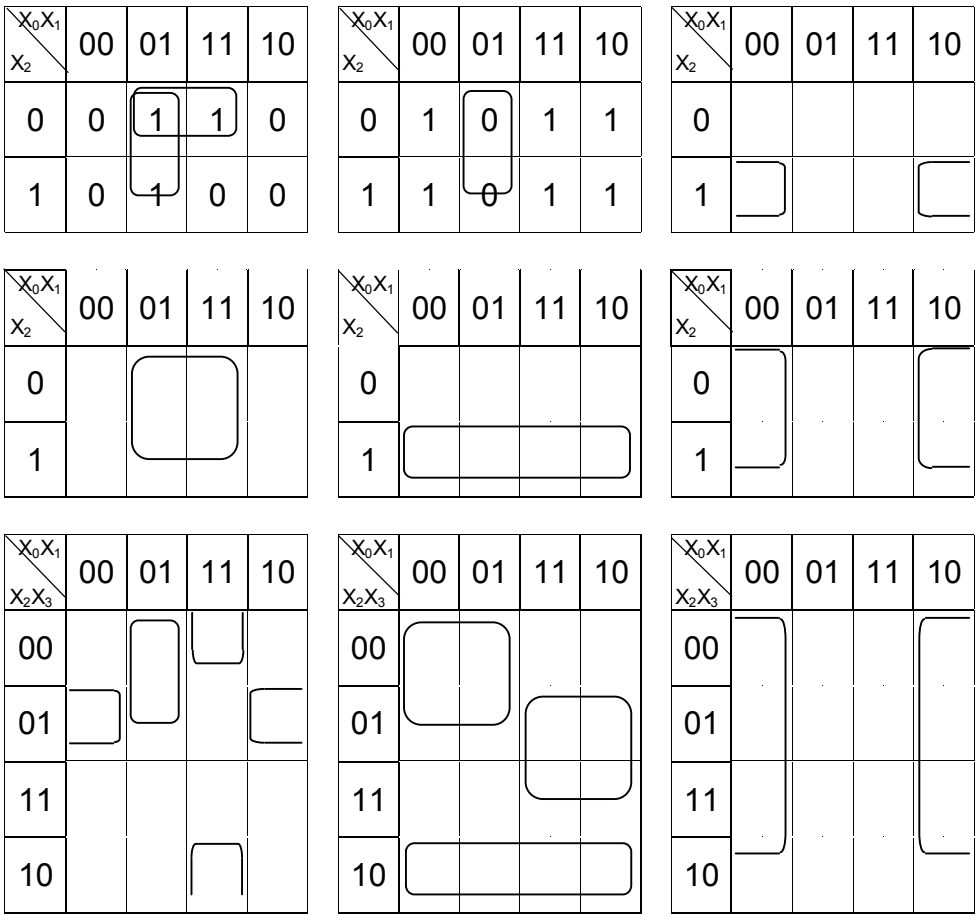
Warto zauważyć fakt, że wypisane na krawędziach tablicy Karnaugha liczby nie są kolejnymi liczbami dwójkowymi lecz kolejnymi słowami kodu Gray'a, a więc kolejne słowa różnią się tylko na jednej pozycji. Kolejność taka jest charakterystyczną własnością tablicy Karnaugha, wykorzystywaną do przeprowadzania uproszczeń w oparciu o tzw. regułę sklejania:

$AX + A\overline{X} = A$ lub $(B + X)(B + \overline{X}) = B$

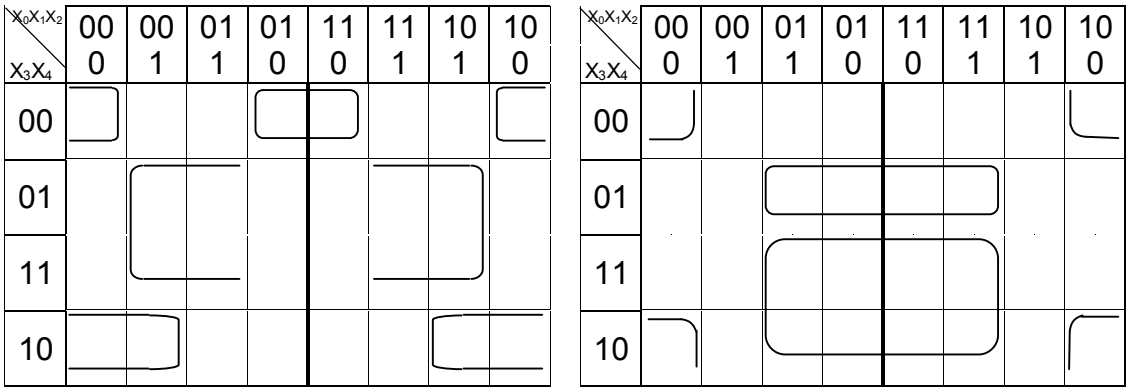
Tak więc zmienną która przyjmuje różne wartości w dwóch sąsiednich polach można pominąć.
Proces minimalizacji funkcji logicznej za pomocą tabeli Karnaugha składa się z trzech etapów. Pierwszy etap polega na przygotowaniu tablicy dla danej liczby zmiennych i wpisaniu w pola elementarne wartości funkcji. W polach odpowiadających kombinacjom zmiennych, dla których wartość funkcji jest nieokreślona, należy wpisać znak nieokreśloności np. ' – '.
Następnie należy narysować obwiednie możliwie największych obszarów obejmujących wyłącznie jedynki (dla postaci sumacyjnej), albo wyłącznie zera (dla postaci iloczynowej), sąsiadujące ze sobą.
Rysowanie obwiedni odbywa się według następujących zasad:

- liczba pól elementarnych połączonych ze sobą musi byś potęgą dwójki (1, 2, 4, ... , 2^n)
- łączone pola muszą być sąsiednimi tzn. oddzielonymi od siebie linią pionową lub poziomą albo krawędzią tablicy
- łączone pole musi mieć kształt symetryczny względem swoich osi

Istnieje jeden wyjątek od powyżej przedstawionych zasad: w dużych tabelach, zawierających pięć lub więcej zmiennych, łączenia można dokonywać nie tylko między sąsiadującymi polami, lecz między wszystkimi takimi wierszami albo kolumnami dla których słowa kodu Gray'a różnią się tylko jednym bitem.
Przykłady łączeń dla trzech i czterech zmiennych przedstawia rysunek 1.2, rysunek 1.3 przedstawia przykładowe łączenia w tabeli z pięcioma zmiennymi.



rys. 1.2



rys 1.3

Z pozoru niewłaściwe połączenie kolumn 001, 011, 111 i 101 (rys. 1.3 – po lewej) jest możliwe gdyż kolumny 011 i 111 różnią się między sobą tylko jednym bitem kodu Gray'a.

Jeśli w tabeli występują znaki nieokreśloności, to pola elementarne, w których występuje ten znak, można (ale nie trzeba!) łączyć z jedynkami bądź zerami zachowując zasady łączenia przedstawione powyżej. Łączenie takie ma na celu uproszczenie funkcji logicznych, więc najlepiej przeprowadzać je tak, aby zakreślony obszar (zawierający np. jedynki i znaki nieokreśloności) był jak największy. Im większy obszar zakreślony w tablicy Karnaugha, tym mniej zmiennych będzie zawierać opisująca go funkcja logiczna. Przykład zastosowania tablic Karnaugha zawierających znaki nieokreśloności został opisany w punkcie 2.1.2.

Ostatnią czynnością wykonywaną podczas minimalizowania funkcji logicznej powyższą metodą jest odczytanie postaci funkcji bezpośrednio z tabeli Karnaugha i zapisanie jej analitycznie za pomocą funkcji boolowskich (dodatek A) w postaci jednej z dwóch form kanonicznych: pełnych iloczynów lub pełnych sum.

Oto kilka przykładów :

przykład 1.1

AB / C	00	01	11	10
0	0	1 (2)	0	0
1	1 (1)	1	1 (3)	0

AB / C	00	01	11	10
0	0 (1)	1	0 (3)	0
1	1	1	1	0 (2)

Postać **pełnych iloczynów** otrzymujemy łącząc w tabeli pola zawierające jedynki (tabela po lewej). Uwzględniamy tylko te zmienne wejściowe, które dla wszystkich elementów grupy mają taką samą wartość. Zmienną wejściową przepisujemy w postaci niezanegowanej jeśli ma wartość 1 a w postaci zanegowanej jeśli ma wartość 0.
Otrzymujemy:

$$Y = \overline{A}_{1} C + \overline{A}_{2} B + B_{3} C$$

Postać **pełnych sum** tworzymy zaznaczając w tabeli zera (tabela po prawej) i wypisując zmienne wejściowe w postaci zanegowanej jeśli mają wartość 1 a niezanegowanej jeśli mają wartość 0.

$$Y = (B + C)^{1} \cdot (\overline{A} + B)^{2} \cdot (\overline{A} + C)^{3}$$

przykład 1.2

AB / CD	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

AB / CD	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

Postać pełnych iloczynów:

$$Y = B \cdot D$$

Postać pełnych sum:

$$Y = B \cdot (\overline{B} + D)$$

przykład 1.3

ABC / DEF	000	001	011	010	110	111	101	100
000					1	1		
001					1	1		
011	1		1	1		1		1
010					1			
110		1					1	
111	1	1	1	1	1			1
101			1		1		1	
100		1	1		1		1	

Dla zachowania przejrzystości przedstawiliśmy w tabeli tylko jedynki. W pustych miejscach należy wyobrazić sobie zera.
Postępując analogicznie jak w poprzednich przykładach otrzymujemy sympatyczną postać funkcji:

$$Y = A \cdot B \cdot \overline{D} \cdot \overline{E} + \overline{A} \cdot B \cdot C \cdot D \cdot \overline{E} + A \cdot \overline{B} \cdot C \cdot D \cdot \overline{E} + \overline{B} \cdot \overline{C} \cdot E \cdot F + \overline{A} \cdot B \cdot E \cdot F +$$
$$+ A \cdot B \cdot \overline{C} \cdot \overline{D} \cdot \overline{F} + \overline{B} \cdot C \cdot D \cdot \overline{F} + B \cdot C \cdot \overline{D} \cdot E \cdot F + D \cdot E \cdot F + A \cdot B \cdot \overline{C} \cdot \overline{E} +$$
$$+ \overline{A} \cdot D \cdot E \cdot F$$

przykład 1.4

Poniższy przykład ilustruje w jaki sposób można dalej uprościć funkcję logiczną otrzymaną bezpośrednio z tablicy Karnaugha.

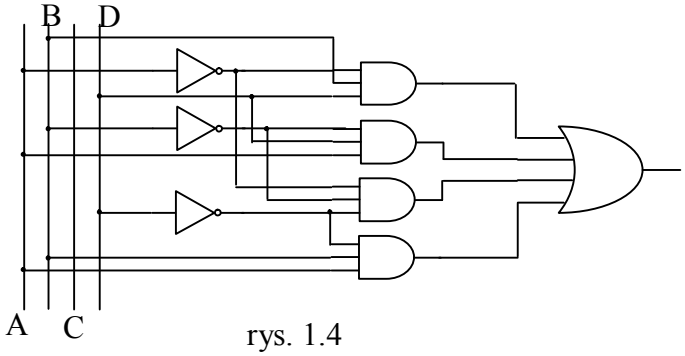
Załóżmy, że mamy następującą tabelę Karnaugha:

CD \ AB	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	0	1	0	1
10	1	0	1	0

Odczytując postać funkcji bezpośrednio z tabeli otrzymujemy:

$$Y = \overline{A} \cdot B \cdot D + A \cdot \overline{B} \cdot D + \overline{A} \cdot \overline{B} \cdot \overline{D} + A \cdot B \cdot \overline{D}$$

Realizacja tej funkcji za pomocą podstawowych bramek logicznych (dodatek B) wygląda następująco (rys. 1.4) :



rys. 1.4

Spróbujmy nieznacznie uprościć postać funkcji korzystając z twierdzeń Boole'a oraz z dwóch poniższych związków:

$A \cdot \overline{B} + \overline{A} \cdot B = A \oplus B$ (Exclusive-OR)

$A \cdot B + \overline{A} \cdot \overline{B} = A \otimes B$ (Exclusive-NOR)

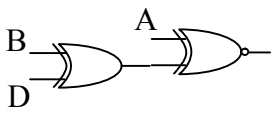
$$Y = \overline{A} \cdot B \cdot D + A \cdot \overline{B} \cdot D + \overline{A} \cdot \overline{B} \cdot \overline{D} + A \cdot B \cdot \overline{D} =$$

$$= \overline{A} \cdot (B \cdot D + \overline{B} \cdot \overline{D}) + A \cdot (\overline{B} \cdot D + B \cdot \overline{D}) =$$

$$= \overline{A} \cdot (B \otimes D) + A \cdot (B \oplus D) =$$

$$= \overline{A} \cdot (\overline{B \oplus D}) + A \cdot (B \oplus D) =$$

$$= A \otimes (B \oplus D)$$



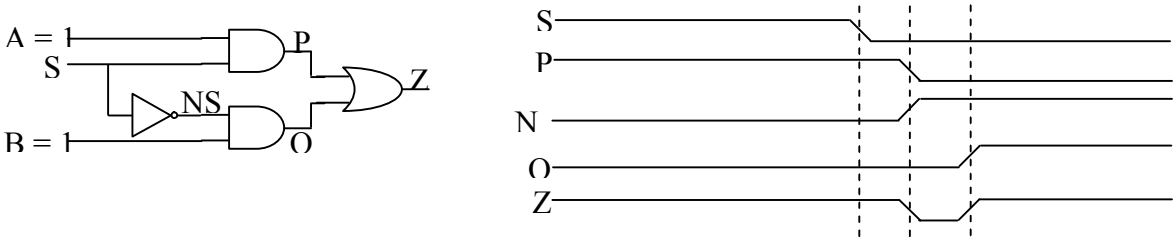
rys. 1.5

Co ilustruje rysunek 1.5.
Jak widać postać funkcji odczytanej z tabeli Karnaugha da się zminimalizować co bywa istotne gdy mamy do wykonania skomplikowany i rozbudowany układ logiczny.

1.2 Hazardy w tablicach Karnaugha i sposoby ich usuwania

Powstawanie hazardów w układach logicznych spowodowane jest opóźnieniami, które wprowadzają poszczególne bramki. Opóźnienie bramki jest inaczej nazywane czasem propagacji. Czas propagacji to czas ustalania sygnału na wyjściu bramki. Czas propagacji jest różny dla różnych bramek: zależy od rodzaju bramki, jej obciążenia na wyjściu, długości ścieżki łączącej bramkę z innymi bramkami oraz od tego czy zmiana występuje ze stanu niskiego na wysoki (t_{LH}) czy wysokiego na niski (t_{HL}). Producenci bramek podają maksymalny gwarantowany czas propagacji dla swoich bramek. Jest to czas propagacji bramki w najgorszym możliwym przypadku. Przy projektowaniu niektórych układów (szczególnie asynchronicznych), dla poprawnej ich pracy, ważne jest również minimalne opóźnienie bramki. Dlatego katalogi podają maksymalne, minimalne i typowe opóźnienie. Warto zwrócić uwagę, że przy symulacji opóźnienie bramki jest idealizowane i z reguły przyjmuje się, że jest ono maksymalne. W rzeczywistości jest ono w zakresie pomiędzy wartością minimalną a maksymalną. W regularnej sieci dwupoziomowej hazardy występują tylko na wskutek różnych czasów propagacji bramek, dlatego w celu wykrycia hazardów należy rozważać równocześnie czasy propagacji maksymalne i minimalne, co nie jest możliwe dla typowej symulacji. Dlatego wykrycie hazardów podczas symulacji jest bardzo trudne – nie występowanie hazardu podczas symulacji nie świadczy, że nie występują one w praktyce. W przypadku symulacji w programie Foundation (na poziomie bramek) należy użyć symulacji w trybie unit, kiedy to wszystkie bramki wnoszą opóźnienie równe 1ns. Aby jednak zaobserwować hazardy należy zróżnicować opóźnienie bramek np. przez zastosowanie dodatkowych inwertorów.

Mechanizm powstawania hazardów zilustrujemy na poniższym przykładzie (rys. 1.6).



rys. 1.6

Gdyby bramki nie miała czasu propagacji, to wyjście Z byłoby cały czas w stanie ’1’. Krótkotrwały spadek poziomu do zera powstał na skutek nierównych czasów propagacji dla różnych ścieżek w układzie. Jeżeli w układzie występują niepożądane szpilki (krótkotrwałe zmiany poziomu) to mówimy, że układ ma problemy z hazardami.
Wyróżniamy hazardy:

- statyczne - kiedy wyjście które ma nie zmienić stanu krótkotrwale zmienia stan na przeciwny
  - w zerze kiedy wyjście ma pozostać w 0
  - w jedynce kiedy wyjście ma pozostać w 1 (rys. 1.6)
- dynamiczne - kiedy występują wielokrotne zmiany stanów na wyjściach (rys. 1.7)

rys. 1.7

Hazardy mogą powodować nieprawidłowe działanie układów jeżeli wyjścia na których się pojawiają są interpretowane asynchronicznie. Na przykład gdy powstaje na sygnale zegarowym, asynchronicznym zerowaniu, itd. W przypadku sygnałów synchronicznych, hazardy (a

właściwie przejściowe stany nieustalone) nie powodują zakłócenia działania układu ponieważ układ próbkuje stan tych sygnałów tylko dla narastającego/ opadającego sygnału zegarowego, kiedy to stan tych wejść jest już ustalony – przyjmuje się, że okres zegara musi być większy od maksymalnego czasu propagacji w układzie (+ stan propagacji od sygnału zegarowego do wyjścia Q + czas ustalania (setup) przerzutnika). Dzięki temu, że hazardy nie występują dla układów synchronicznych, układy synchroniczne stały się bardzo popularne i praktycznie całkowicie wyeliminowały układy asynchroniczne, które stosuje się bardzo rzadko. Układy asynchroniczne stosuje się np. jako asynchroniczny reset (zaleca się stosowanie resetu synchronicznego, a reset asynchroniczny używany jest tylko dla wymuszenia stanu początkowego). Układy asynchroniczne są stosowane również do bramkowania sygnału zegarowego (np. jeżeli sygnał zegarowy ma być aktywny co trzeci sygnał zegarowy wtedy bramkuje się go odpowiednio bramką AND i sygnałem pochodzącym z licznika modulo 3), jednakże zaleca się stopowanie wejścia clock enable (CE), które działa w podobny sposób jak bramkowanie sygnału zegarowego, jednakże nie powoduje powstawania zjawiska wyścigu.

Sposoby usuwania hazardów prześledzimy na przykładzie funkcji logicznej zilustrowanej następująca tablicą Karnaugha (rys. 1.8)

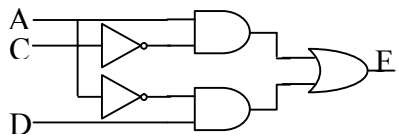
AB / CD	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

rys. 1.8

Na podstawie tablicy Karnaugha otrzymujemy zminimalizowaną funkcję logiczną:

$$F = A \cdot \overline{C} + \overline{A} \cdot D$$

Jej implementacja za pomocą bramek przedstawiona jest na rysunku 1.9



rys. 1.9

Prześledźmy zmianę stanów na wejściach układu 1101 → 0101 (zmiana stanu wejścia A z ‘1’ na ‘0’). Zauważmy, że ‘0’, które podaliśmy na wejście A zostanie podane natychmiast na wejście górnej bramki AND natomiast z opóźnieniem na wejście dolnej bramki AND (opóźnienie to jest wywołane dodatkową bramką negacji). Tak więc przez pewien krótki czas (odpowiadający czasowi propagacji bramki NOT) na wejścia obydwu bramek AND będzie podany stan ‘01’; na ich wyjściach pojawią się więc zera – czyli na wyjściu układu otrzymamy krótkotrwały spadek poziomu do zera - hazard. Po tym czasie, gdy sygnał z wejścia A dotrze na wejście dolnej bramki AND, wyjście układu powróci w stan ‘1’.
Dzieje się tak ponieważ w tablicy Karnaugha komórki ‘0101’ i ‘1100’ sąsiadują ze sobą, lecz nie są objęte wspólną grupą (patrz rys. 1.10). Sposobem wyeliminowania tego hazardu jest

połączenie dodatkową grupą sąsiadujących, lecz nie połączonych ze sobą elementów.
Otrzymamy tym samym następującą funkcję logiczną: $F = A \cdot \overline{C} + \overline{A} \cdot D + \overline{C} \cdot D$
Zaimplementowany układ będzie wolny od hazardów ponieważ przy wyżej opisanej zmianie stanu wyjście będzie podtrzymywane w stanie '1' przez dodatkową bramkę AND ($\overline{C} \cdot D$), która nie zmienia stanu w trakcie tej operacji.

AB / CD	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

rys. 1.10

Ogólny algorytm postępowania dla sieci dwupoziomowej:

- aby usunąć hazardy w postaci sum iloczynów: sprawdzić w tablicy prawdy, czy wszystkie przylegające jedynki są pokryte wspólną grupą (oczkiem w tablicy Karnaugha). Jeżeli nie to należy dodać dodatkowe grupy
- aby usunąć hazardy w postaci iloczynów sum: sprawdzamy w tablicy prawdy czy wszystkie przylegające zera mają wspólne grupy. Jeśli nie to dodajemy dodatkowe grupy.

W sieci dwupoziomowej wyeliminowanie hazardu statycznego gwarantuje wykonanie sieci wolnej od hazardów dynamicznych.

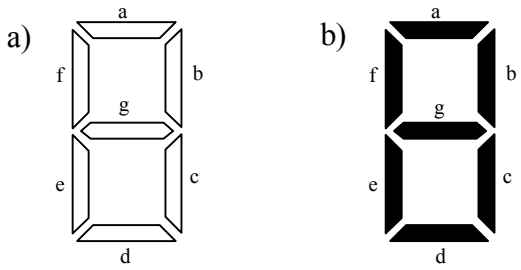
W sieciach wielopoziomowych eliminacja hazardów statycznych nie wystarcza do eliminacji hazardów dynamicznych. Istnieją techniki eliminujące hazardy dynamiczne w sieciach wielopoziomowych, ale są one raczej dość skomplikowane, więc najlepiej jest budować sieci dwupoziomowe.
Warto zwrócić uwagę, że eliminacja hazardów zakłada, że zmianie ulega równocześnie nie więcej niż jeden sygnał wejściowy. Praktycznie jest bardzo mało prawdopodobne, że zmieniają się w tym samym czasie (z dokładnością co do propagacji pojedynczej bramki – około 1ns) dwa lub więcej sygnałów, dlatego założenie to jest prawie zawsze spełnione (chyba, że układ asynchroniczny jest sterowny układem synchronicznym). Warto jednak zwrócić uwagę, że przy symulacji możemy łatwo w tym samym czasie zmieniać wszystkie sygnały wejściowe (co nie jest możliwe w praktyce) i dlatego ciągle mogą być obserwowane hazardy, mimo tego, że w teorii one nie występują.

UKŁADY KOMBINACYJNE – przykłady

W układach kombinacyjnych, w odróżnieniu od sekwencyjnych, wektor stanów wyjściowych uzależniony jest jedynie od wektora stanów wejściowych i kombinacja stanów na wyjściu jest jednoznacznie zdeterminowana przez kombinację stanów na wejściu. Układy te nie posiadają pamięci.
W dalszej części przedstawimy szereg przykładów zastosowań układów kombinacyjnych.

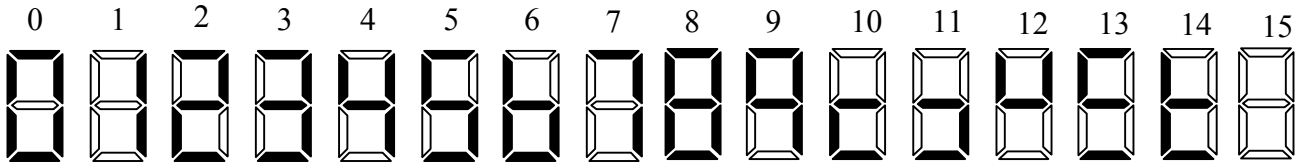
2.1.1 Układ sterujący siedmiosegmentowym wyświetlaczem
(transkoder kodu BCD na kod 7-segmentowego wskaźnika)

Na rysunku 2.1 przedstawiliśmy wyświetlacz siedmiosegmentowy ze wszystkimi blokami wyłączonymi (a) oraz włączonymi (b). Litery przy poszczególnych blokach są ogólnie przyjętymi umownymi oznaczeniami.



rys. 2.1

Rozpatrywany transkoder ma 4 wejścia (A, B, C, D), a więc liczba możliwych stanów wejściowych wynosi 16. Stany 0 – 9 odpowiadają wyświetleniu cyfr a 6 pozostałych stanów powoduje wyświetlenie znaków dodatkowych. Zestaw wyświetlanych cyfr i znaków przedstawiliśmy na rysunku 2.2.



rys. 2.2

Ten zestaw odpowiada standardowemu zestawowi znaków jaki zaimplementowano w układzie scalonym UCY7447N.

Tablicę prawdy dla tego transkodera pokazaliśmy na rysunku 2.3.

	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	0	0	0	1	1	0	1
11	1	0	1	1	0	0	1	1	0	0	1
12	1	1	0	0	0	1	0	0	0	1	1
13	1	1	0	1	1	0	0	1	0	1	1
14	1	1	1	0	0	0	0	1	1	1	1
15	1	1	1	1	0	0	0	0	0	0	0

rys. 2.3

Funkcje logiczne dla siedmiu wyjść a, b, c, d, e, f, g mogą być wyznaczone z tablicy prawdy. Można to uczynić przez wypisanie stanów wejściowych dla których rozpatrywane wyjście przyjmuje stan 1, tzn. rozpatrywany segment jest włączony.
Aby sporządzić tablicę Karnaugha np. dla segmentu ‘a’ zauważmy, że jest on włączony gdy wyświetlane są znaki o numerach: 0, 2, 3, 5, 7, 8, 9, 13.
Rozumując analogicznie:

- segment ‘b’ jest włączony dla 0, 1, 2, 3, 4, 7, 8, 9, 12
- segment ‘c’ dla: 0, 1, 3, 5, 6, 7, 8, 9, 11
- segment ‘d’ dla: 0, 2, 3, 5 ,6, 8, 10, 11, 13, 14
- segment ‘e’ dla: 0, 2, 6, 8, 10, 14
- segment ‘f’ dla: 0, 4, 5, 6, 8, 9, 12, 13, 14
- segment ‘g’ dla: 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14

Tablice Karnaugha dla wszystkich funkcji są przedstawione na rysunkach 2.4 i 2.5.

blok a

BA / DC	00	01	11	10
00	1_0	0_1	1_3	1_2
01	0_4	1_5	1_7	0_6
11	0_{12}	1_{13}	0_{15}	0_{14}
10	1_8	1_9	0_{11}	0_{10}

blok b

BA / DC	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	1	0	0	0
10	1	1	0	0

blok c

BA / DC	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	0

rys. 2.4

blok d

BA / DC	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	0	1	0	1
10	1	0	1	1

blok e

BA / DC	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	0	0	0	1
10	1	0	0	1

blok f

BA / DC	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	1	1	0	1
10	1	1	0	0

blok g

BA / DC	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	1	1	0	1
10	1	1	1	1

rys. 2.5

Korzystając z tablic Karnaugha otrzymaliśmy następujące funkcje:

$a = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot D + B \cdot \overline{C} \cdot \overline{D} + A \cdot C \cdot \overline{D}$

$b = \overline{A} \cdot \overline{B} + \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C}$

$c = C \cdot \overline{D} + A \cdot B + \overline{B} \cdot \overline{C}$

$d = \overline{A} \cdot \overline{C} + \overline{A} \cdot B + B \cdot \overline{C} + A \cdot \overline{B} \cdot C$

$e = B \cdot \overline{A} + \overline{C} \cdot \overline{A}$

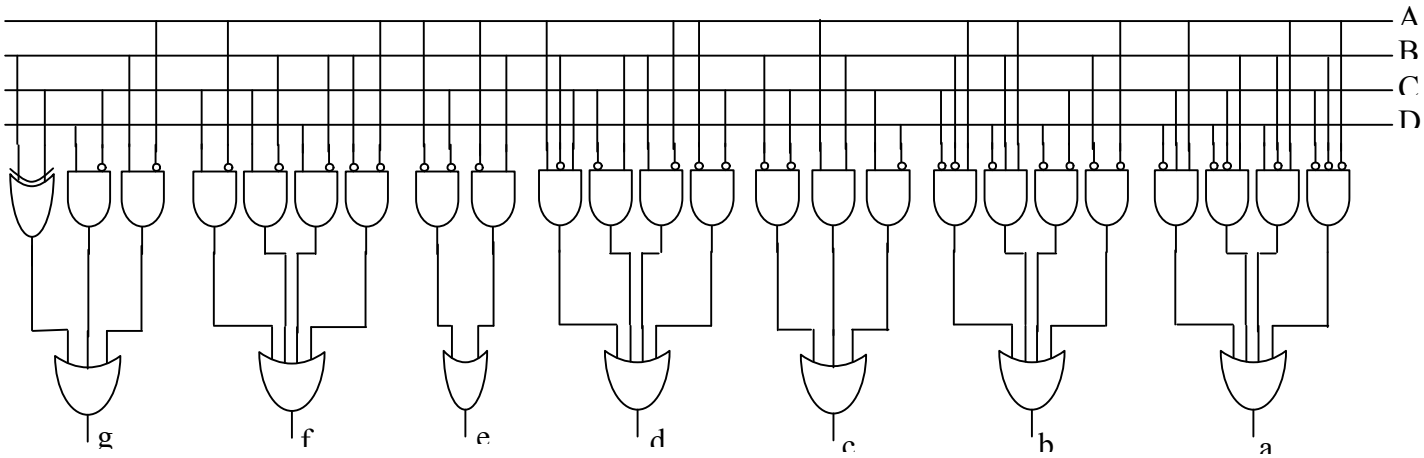
$f = \overline{A} \cdot \overline{B} + \overline{B} \cdot \overline{C} \cdot D + C \cdot \overline{B} + \overline{A} \cdot C =$

$= \overline{A} \cdot \overline{B} + \overline{B} \cdot (\overline{C} \cdot D + C) + \overline{A} \cdot C =$

$= \overline{A} \cdot \overline{B} + \overline{B} \cdot (D + C) + \overline{A} \cdot C =$

$= \overline{A} \cdot \overline{B} + \overline{B} \cdot D + \overline{B} \cdot C + \overline{A} \cdot C$

$g = \overline{A} \cdot B + \overline{C} \cdot D + \overline{B} \cdot C + B \cdot \overline{C} = \overline{A} \cdot B + \overline{C} \cdot D + B \oplus C$



rys. 2.6 – schemat logiczny transkodera

2.1.2 Układ sterujący siedmiosegmentowym wyświetlaczem – zastosowanie tablic

Karnaugha ze znakami nieokreśloności.

Rozważmy układ analogiczny do poprzedniego, tyle że tym razem interesuje nas jedynie wyświetlenie cyfr 0 – 9, a pozostałe stany są nieokreślone. Zmodyfikowana tablica prawdy z rysunku 2.3 jest przedstawiona na rysunku 2.3a

	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

2.3a

Tablice Karnaugha tego układu będą zawierać znaki nieokreśloności (‘–‘). Na rysunku 2.4a przedstawiono te tablice dla bloków ‘a’ i ‘b’.

blok a

BA / DC	00	01	11	10
00	1_0	0_1	1_3	1_2
01	0_4	1_5	1_7	0_6
11	$-_{12}$	$-_{13}$	$-_{15}$	$-_{14}$
10	1_8	1_9	$-_{11}$	$-_{10}$

blok b

BA / DC	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	-	-	-	-
10	1	1	-	-

rys. 2.4a

Zasadę postępowania z takimi tablicami opisano w punkcie 1.1. Po minimalizacji otrzymujemy następujące funkcje logiczne:

$a = D + A \cdot C + A \cdot B + \overline{A} \cdot \overline{C} = A \cdot B + D + A \otimes C$
$b = \overline{C} + \overline{A} \cdot \overline{B} + A \cdot B = \overline{C} + \overline{A} \otimes \overline{B}$

Postępowanie dla pozostałych bloków wyświetlacza przebiega w sposób analogiczny.

2.2 Transkoder przetwarzający naturalny czterobitowy kod dwójkowy na kod 8421 BCD.
Za pomocą czterech bitów wejścia możemy zakodować liczby od 0 do 15, zatem do zakodowania wyjścia potrzebujemy tylko pięciu bitów – czterech bitów na cyfrę jedności i jednego bitu na cyfrę dziesiątek.

Tablica prawdy tego transkodera jest przedstawiona na rysunku 2.7.

	D	C	B	A	A”	D’	C’	B’	A’
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0
3	0	0	1	1	0	0	0	1	1
4	0	1	0	0	0	0	1	0	0
5	0	1	0	1	0	0	1	0	1
6	0	1	1	0	0	0	1	1	0
7	0	1	1	1	0	0	1	1	1
8	1	0	0	0	0	1	0	0	0
9	1	0	0	1	0	1	0	0	1
10	1	0	1	0	1	0	0	0	0
11	1	0	1	1	1	0	0	0	1
12	1	1	0	0	1	0	0	1	0
13	1	1	0	1	1	0	0	1	1
14	1	1	1	0	1	0	1	0	0
15	1	1	1	1	1	0	1	0	1

rys. 2.7

Zauważmy, że kolumna A’ jest tożsama z kolumną A (A’ = A).
Funkcje logiczne opisujące pozostałe wyjścia otrzymamy za pomocą tablic Karnaugha (rys. 2.8).

B’

BA / DC	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	0	0
10	0	0	0	0

C’

BA / DC	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	1	1
10	0	0	0	0

D’

BA / DC	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

A”

BA / DC	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$B' = B \cdot \overline{D} + \overline{B} \cdot C \cdot D$
$C' = C \cdot \overline{D} + B \cdot C$
$D' = \overline{B} \cdot \overline{C} \cdot D$
$A'' = B \cdot D + C \cdot D$

rys. 2.8

2.3 Transkoder służący do przetwarzania kodu 8421 BCD na kod 2*421 BCD

W celu zaprojektowania takiego transkodera należy przedstawić rozpatrywane kody w tablicy (rysunek 2.9) . W kodzie 2*421 BCD cyfry od 0 do 7 są kodowane w sposób naturalny, natomiast cyfry 8 i 9 jako 8 = 2 + 6 i 9 = 2 + 7 (dodatkowa dodawana dwójka jest kodowana na pierwszym bicie). Nazwa kodu pochodzi od wag poszczególnych bitów.

waga	D 8	C 4	B 2	A 1	D' 2	C' 4	B' 2	A' 1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	0	1	0	1
6	0	1	1	0	0	1	1	0
7	0	1	1	1	0	1	1	1
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1

rys. 2.9

Zauważmy, że kolumny wejścia A i D są tożsame z kolumnami wyjścia A' i D' czyli
A' = A i D' = D

W celu wyznaczenia funkcji B' i C' skorzystamy z tablic Karnaugha (rys. 2.10)

B'

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	-	-	-	-
10	1	1	-	-

C'

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	-	-	-	-
10	1	1	-	-

B' = B + D
C' = C + D

rys. 2.10

2.4 Komparator liczb dwubitowych.

Zaprojektujemy komparator realizujący funkcję A > B (jeśli A>B – dostajemy na wyjściu 1, w przeciwnym przypadku – otrzymujemy 0). Dwubitowe liczby A i B będą przedstawione w postaci $A = A_2A_1$ i $B = B_2B_1$ (A_2, A_1, B_2, B_1 są wejściami układu).
Tablica prawdy dla tego komparatora i tablica Karnaugha są przedstawione na rysunku 2.11.

	A_2	A_1	B_2	B_1	A>B
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

A_2A_1 \ B_2B_1	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

rys. 2.11

Otrzymujemy następującą funkcję logiczną:

$$A > B = \overline{B}_2 \cdot A_2 + \overline{B}_1 \cdot \overline{B}_2 \cdot A_1 + A_1 \cdot A_2 \cdot \overline{B}_1$$

2.5 Układ sprawdzający czy liczba od 0 do 15 jest podzielna przez 3.

Na wejście tego układu podajemy liczbę zapisaną w naturalnym kodzie dwójkowym – na wyjściu otrzymujemy 1 bądź 0 w zależności czy liczba jest (1) czy też nie jest (0) podzielna przez 3.

	D	C	B	A	W
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

DC \ BA	00	01	11	10
00	1	0	1	0
01	0	0	0	1
11	1	0	1	0
10	0	1	0	0

rys. 2.12 – tablica prawdy i tablica Karnaugha

$$W = A \cdot \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot D + A \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot \overline{D} + A \cdot B \cdot C \cdot D + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} =$$
$$= \overline{B} \cdot D \cdot (A \cdot \overline{C} + \overline{A} \cdot C) + B \cdot \overline{D} \ (A \cdot \overline{C} + \overline{A} \cdot C) + A \cdot B \cdot C \cdot D + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} =$$

$= (A \oplus C) \cdot (B \oplus D) + A \cdot B \cdot C \cdot D + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$

2.6 Układ kombinacyjny sterujący napełnianiem basenu kąpielowego.

Rozważmy w jaki sposób zaprojektować układ kombinacyjny nadzorujący napełnianie basenu.

Dopływ wody jest sterowany zaworem X, a jej odpływ zaworem Y. Podanie ‘1’ do układu sterowania zaworu oznacza otwarcie zaworu a podanie ‘0’ - zamknięcie. W basenie znajdują się trzy czujniki: A, B, C (wejścia układu) wyznaczające odpowiednio maksymalny, średni i minimalny poziom wody. Zadziałanie czujnika następuje po zanurzeniu go w wodzie i jest sygnalizowane pojawieniem się jedynki na jego wyjściu. Lustro wody nie powinno obniżać się poniżej poziomu minimalnego. Dodatkowe wejście Z powoduje włączenie układu alarmu w przypadku uszkodzenia któregoś z czujników (jeśli np. czujnik B pokazuje ‘1’ a czujnik C, niższy, pokazuje wciąż ‘0’. Jednocześnie z sygnałem alarmu następuje zamknięcie zaworu dopływu i otwarcie zaworu odpływu wody. Między stanami wody średnim i maksymalnym powinny być otwarte oba zawory w celu ciągłej wymiany wody w basenie.

Na podstawie takiego opisu słownego można sporządzić tablicę wartości oraz tablice Karnaugha dla wyjść X, Y, Z (rysunek 2.13).

	A	B	C	X	Y	Z
0	0	0	0	1	0	0
1	0	0	1	1	0	0
2	0	1	0	0	1	1
3	0	1	1	1	1	0
4	1	0	0	0	1	1
5	1	0	1	0	1	1
6	1	1	0	0	1	1
7	1	1	1	0	1	0

X

AB / C	00	01	11	10
0	1	0	0	0
1	1	1	0	0

Y

AB / C	00	01	11	10
0	0	1	1	1
1	0	1	1	1

Z

AB / C	00	01	11	10
0	0	1	1	1
1	0	0	0	1

rys. 2.13

Otrzymujemy następujące zminimalizowane funkcje logiczne:

$X = \overline{A} \cdot \overline{B} + \overline{A} \cdot C$

$Y = A + B$

$Z = B \cdot \overline{C} + A \cdot \overline{C} + A \cdot \overline{B}$

2.7 Enkoder służący do zamiany kodu 1 z 10 na kod BCD 8421.

Enkoderami nazywane są układy służące do konwersji kodu 1 z n na określony kod wyjściowy. Układy te mają więc n wejść przy czym w danym momencie tylko jedno z wejść jest wyróżnione (stan ‘1’). Na wyjściach enkodera pojawia się numer wejścia wyróżnionego przedstawiony w żądanym kodzie dwójkowym. Enkodery są stosowane głównie do wprowadzania informacji z przełączników dziesięciopozycyjnych obrotowych lub klawiszowych. Wejścia układu oznaczyliśmy symbolami I_0 do I_9.

Tablica prawdy opisująca działanie enkodera 1 z 10 na 8421 BCD jest przedstawiona na rysunku 2.14.

Kod 1 z 10										Kod 8421 BCD			
I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	D	C	B	A
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

rys. 2.14

Korzystając bezpośrednio z tabeli prawdy można zapisać następujące funkcje logiczne opisujące wyjścia układu (zakładając, że w danym momencie tylko na jednym wejściu jest stan ‘1’):

$A = I_1 + I_3 + I_5 + I_7 + I_9$ $\quad B = I_2 + I_3 + I_6 + I_7$
$C = I_4 + I_5 + I_6 + I_7$ $\quad D = I_8 + I_9$

2.8 Dekoder kodu 8421 BCD na kod 1 z 10

Dekoderem nazywamy układ kombinacyjny służący do konwersji kodu dwójkowego na kod 1 z n (patrz przykład 2.7).
Wyjścia układu oznaczyliśmy symbolami I_0 do I_9. Dane wyjście jest w stanie wyróżnionym (‘1’) jeżeli na wejściach A, B, C, D jest odpowiadająca mu kombinacja stanów logicznych kodu 8421. W przypadku podania na wejścia niewykorzystanych w kodzie stanów logicznych (np. 1111) wszystkie wyjścia są w stanie ‘0’.Tablicę prawdy rozważanego dekodera przedstawiono na rysunku 2.15.

Kod 8421 BCD				Kod 1 z 10									
D	C	B	A	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0
0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
0	0	0	1	0	0	0	0	0	0	0	0	**1**	0
0	0	1	0	0	0	0	0	0	0	0	**1**	0	0
0	0	1	1	0	0	0	0	0	0	**1**	0	0	0
0	1	0	0	0	0	0	0	0	**1**	0	0	0	0
0	1	0	1	0	0	0	0	**1**	0	0	0	0	0
0	1	1	0	0	0	0	**1**	0	0	0	0	0	0
0	1	1	1	0	0	**1**	0	0	0	0	0	0	0
1	0	0	0	0	**1**	0	0	0	0	0	0	0	0
1	0	0	1	**1**	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0

rys. 2.15

Korzystając bezpośrednio z tablicy prawdy otrzymujemy następujące funkcje:

$I_0 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$ $\quad$ $I_1 = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$

$I_2 = \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D}$ $\quad$ $I_3 = A \cdot B \cdot \overline{C} \cdot \overline{D}$

$I_4 = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D}$ $\quad$ $I_5 = A \cdot \overline{B} \cdot C \cdot \overline{D}$

$I_6 = \overline{A} \cdot B \cdot C \cdot \overline{D}$ $\quad$ $I_7 = A \cdot B \cdot C \cdot \overline{D}$

$I_8 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D$ $\quad$ $I_9 = A \cdot \overline{B} \cdot \overline{C} \cdot D$

2.9 Układ służący do podnoszenia do kwadratu liczb od 0 do 7

Rozważany układ podnosi do kwadratu liczby od 0 do 7 zapisane w naturalnym kodzie dwójkowym i zwraca wynik na 6-bitowe wyjście również w naturalnej postaci dwójkowej. Tablica prawdy i tablice Karnaugha dla tego układu są przedstawione na rys. 2.16.

Liczba	C	B	A	wynik	F'	E'	D'	C'	B'	A'
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	4	0	0	0	1	0	0
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

A’

BA / C	00	01	11	10
0	0	1	1	0
1	0	1	1	0

C’

BA / C	00	01	11	10
0	0	0	0	1
1	0	0	0	1

D’

BA / C	00	01	11	10
0	0	0	1	0
1	0	1	0	0

E’

BA / C	00	01	11	10
0	0	0	0	0
1	1	1	1	0

F’

BA / C	00	01	11	10
0	0	0	0	0
1	0	0	1	1

rys. 2.16

Z tablic otrzymaliśmy następujące funkcje logiczne:
A’ = A
B’ = 0
C’ = $\overline{A} \cdot B$
D’ = $A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C = A \cdot (B \oplus C)$
E’ = $C \cdot \overline{B} + C \cdot A$
F’ = $B \cdot C$

2.10 Układ obliczający pierwiastek kwadratowy z liczby od 0 do 15.

Rozważmy układ służący do obliczenia pierwiastka z liczby od 0 do 15 podanej w naturalnym czterobitowym kodzie dwójkowym.
Wyjście składa się z trzech bitów : na dwóch (O_1, O_2) zapisujemy wynik pierwiastkowania jeśli jest liczbą naturalną; w przeciwnym wypadku zwracany jest błąd – stan ‘1’ na trzecim bicie wyjścia (E) i stan ‘0’ na pozostałych bitach wyjścia.
Tablica prawdy oraz tablice Karnaugha dla tego przykładu są pokazane na rysunku 2.17.

liczba	D	C	B	A	wynik	O_2	O_1	E
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	1	0
2	0	0	1	0	E	0	0	1
3	0	0	1	1	E	0	0	1
4	0	1	0	0	2	1	0	0
5	0	1	0	1	E	0	0	1
6	0	1	1	0	E	0	0	1
7	0	1	1	1	E	0	0	1
8	1	0	0	0	E	0	0	1
9	1	0	0	1	3	1	1	0
10	1	0	1	0	E	0	0	1
11	1	0	1	1	E	0	0	1
12	1	1	0	0	E	0	0	1
13	1	1	0	1	E	0	0	1
14	1	1	1	0	E	0	0	1
15	1	1	1	1	E	0	0	1

O_2

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	1	0	0

O_1

DC \ BA	00	01	11	10
00	0	1	0	0
01	0	0	0	0
11	0	0	0	0
10	0	1	0	0

E

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	1	1	1
11	1	1	1	1
10	1	0	1	1

rys. 2.17

Otrzymujemy:

$O_2 = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D$

$O_1 = A \cdot \overline{B} \cdot \overline{C}$

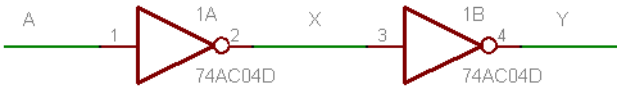
$E = B + A \cdot C + \overline{A} \cdot D$

PRZEBIEG ĆWICZENIA.

1.

1a. uzupełnij tabelę

A	X	Y
0		
+5V		



1b. uzupełnij tabelę używając generatora przebiegów prostokątnych

X	+5 0									

Y	+5 0									

2.

2a. uzupełnij tabelę

Dla pinu 1 podłączonego do +5V	
A	X
0	
5	

2b. uzupełnij tabelę

Dla pinu 1 podłączonego do masy	
A	X
0	
5	

2c. uzupełnij tabelę używając generatora przebiegów prostokątnych dla pinu 1 podłączonego do +5V

A									
X									

2c'.uzupełnij tabelę używając generatora przebiegów prostokątnych dla pinu 1 podłączonego do masy

A									
X									

2d.uzupełnij tabelę

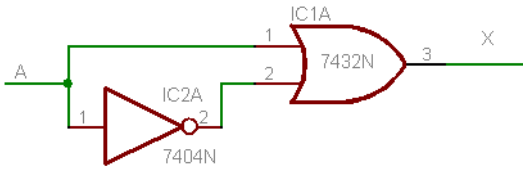
A	Y
0	
5	

2e. uzupełnij tabelę używając generatora przebiegów prostokątnych

A									
X									

2f. uzupełnij tabelę

A	X
0	
5	



2g. uzupełnij tabelę używając generatora przebiegów prostokątnych

A									
X									

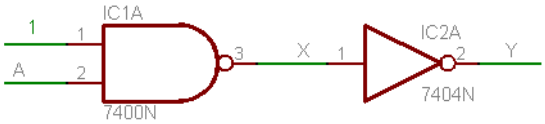
3.

3a.uzupełnij tabelę

Dla pinu 1 podłączonego do +5V	X	Y
A		
0		
+5V		

3b.uzupełnij tabelę

Dla pinu 1 podłączonego do masy	X	Y
A		
0		
+5V		



3c. uzupełnij tabelę używając generatora przebiegów prostokątnych dla pinu 1 podłączonego do +5V

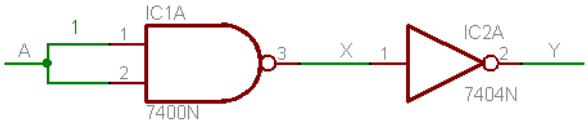
A									
X									
Y									

3c. uzupełnij tabelę używając generatora przebiegów prostokątnych dla pinu 1 podłączonego do masy

A									
X									
Y									

3d.uzupełnij tabelę

A		
0		
+5V		

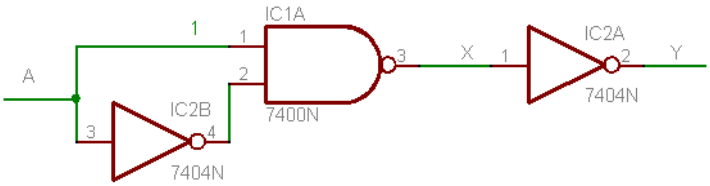


3e. uzupełnij tabelę używając generatora przebiegów prostokątnych

A									
X									
Y									

3f.uzupełnij tabelę

A		
0		
+5V		



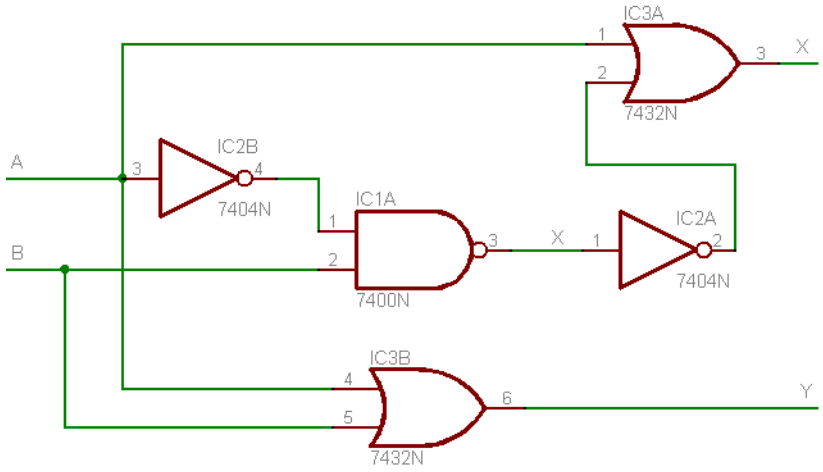
3g. uzupełnij tabelę używając generatora przebiegów prostokątnych

A									
X									
Y									

4.

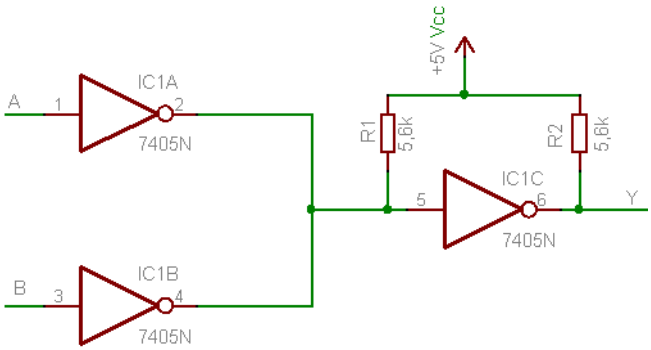
4a. uzupełnij tabelę

A	B	$A\overline{\ }B$	X	Y
0	0			
0	+5V			
+5V	0			
+5V	+5V			



4b. uzupełnij tabelę

A	B	X	Y
0	0		
0	+5V		
+5V	0		
+5V	+5V		



OPRACOWANIE.

Dla części 1, 2 i 3 porównaj ze sobą wartości zmiennej wejściowej A z wartościami Y.
Y może być równe 0 1 lub A Wpisz wartości Y do tabeli 2-12d w postaci 1, 0 lub A. Pomiń krótkie impulsy pojawiające się na wyjściu układów podczas badania układu w warunkach dynamicznych z wykorzystaniem generatora fali prostokątnej (jakie są tego przyczyny). Dla przykładu: w części 1(a) i 1(b) Y zawsze przyjmuje wartość A, więc w rubryce Y wpisujemy A; w części 2(a) i 2(c), Y zawsze równa się 1, w kolumnę Y wpisujemy wartość 1.
2b.Do tabeli 2-12d wpisz równania logiczne które realizują badane układy logiczne np. dla części 1(a) i 1(b) otrzymaliśmy Y=A. Ze schematu logicznego wynika iż układ realizuje podwójną inwersję w takim razie równanie logiczne przyjmie postać $\overline{\overline{A}}$=A.
3. Dla części 4(a)
a). wypisz wartości zmiennych wejściowych dla bramki G1 dla wszystkich kombinacji zmiennych A i B.
b). wpisz wartości X dla wszystkich kombinacji zmiennych A i B.
c). wpisz wartości Y dla wszystkich kombinacji zmiennych A i B
d). w oparciu o przeprowadzony eksperyment co możesz powiedzieć o relacjach między X i Y w rozumieniu algebry Boolea.
e) narysuj tablicę Karnaugh dla wejścia 2 bramki G1 i otrzymany wynik porównaj ze stanami wejściowymi dla bramki G2.
4. Dla części 4(b). Przedstawiony schemat pokazuje bezpośredni sposób łączenie ze sobą kilku wyjść cyfrowych układów logicznych dla osiągnięcia tzw. logiki połączonego OR lub implikowanego AND. Takie połączenie jest dozwolone tylko dla układów wyposażonych w wyjściem z otwartym kolektorem (takie wyjście posiada użyty w doświadczeniu układ 7405).
Układy typu open collector wymagają stosowania tzw. rezystora podciągającego, który włączamy między napięcie zasilania układu a jego wyjście. Można łączyć ze sobą dwa lub więcej wejść używając jednego opornika podciągającego. Podstawowym zastosowaniem tego typu elementów jest proste tworzenie układów OR/NOR bez potrzeby używania dodatkowych bramek OR lub NOR.

CZĘŚĆ		Y WARTOŚĆI A,1 lub 0	Równanie Boolea między wejściem a wyjściem
1a i 1b		A	$A=\overline{\overline{A}}$
2a i 2c	pin1 do +5V	1	A+1=A
2b i 2c	pin1 masy		
2d i 2e			
2f i 2g			
3a i 3c	pin1 do +5V		
3b i 3c	pin1 masy		
3d i 3e			
3f i 3g			

tabela 2-12d